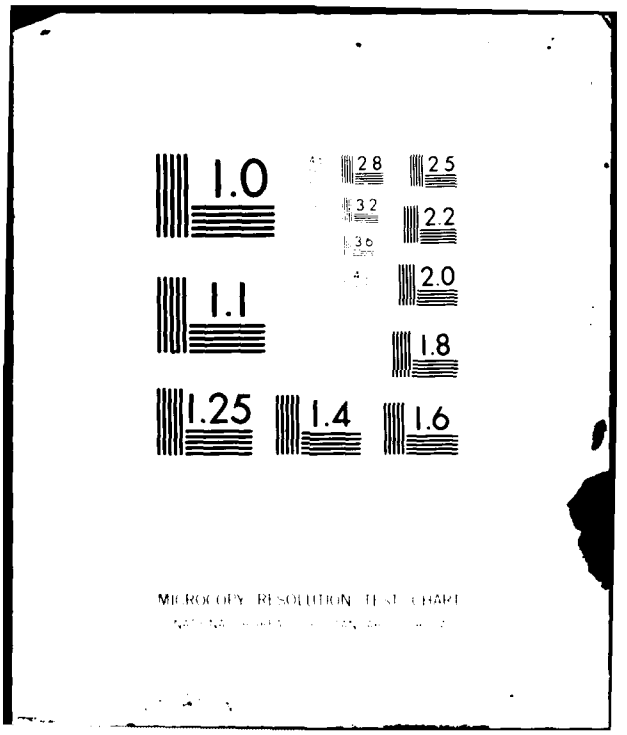
AD-A097 239 MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE F/G 9/2
RESEARCH REPORT FOR AIR FORCE OFFICE OF SCIENTIFIC RESEARCH CON--ETC(U)
DEC 80 V R BASILI, J D GANNON, R G HAMLET F49620-80-C-0001

UNCLASSIFIED AFOSR-TR-81-0315 NL

END
DATE FILMED
5-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
MICROCOPY RESOLUTION TEST CHART

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LEVEL II

12

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
AFOSR-TR- 81-0315	AD-A097 239	

4. TITLE (and Subtitle)
RESEARCH REPORT FOR AIR FORCE OFFICE OF SCIENTIFIC RESEARCH, Contract F49620-80-C-0001

5. TYPE OF REPORT & PERIOD COVERED
INTERIM
1 Jan 80 - 31 Dec 80

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Raymond T. Yeh, Victor R. Basili, John D. Gannon, Richard G. Hamlet and Marvin V. Zelkowitz

8. CONTRACT OR GRANT NUMBER(s)
F49620-80-C-0001

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Department of Computer Science
University of Maryland
College Park MD 20742

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
2304/A2 61102F

11. CONTROLLING OFFICE NAME AND ADDRESS
Air Force Office of Scientific Research/NM
Bolling AFB DC 20332

12. REPORT DATE
31 DEC 80

13. NUMBER OF PAGES
6

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

DTIC ELECTE S APR 2 1981 D D

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report summarizes research supported by contract F49620-80-C-0001 from the Air Force Office of Scientific Research to the University of Maryland for the period 1 Jan 80 to 31 Dec 80. It includes research supported in part for principal investigators Drs. Victor R. Basili, John D. Gannon, Richard Hamlet, Raymond Yeh and Marvin V. Zelkowitz, and senior personnel Drs. Mark Weiser and Pamela Zave.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(18) AFOSR-(19) TR-81-0315

(6) RESEARCH REPORT FOR
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Contract F49620-80-C-0001

(9) Interim Rept. 1 Jan -31 Dec 80

(10) Principal Investigators: Victor R. Basili
John D. Gannon
Richard G. Hamlet
Raymond T. Yeh
Marvin V. Zelkowitz

and

Senior Personnel: I. Miyamoto
Mark Weiser
Pamela Zave

(11) 31 Dec 80

(12) 9

(15) F49620-80-C-0001

(16) 2304

(17) A2

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

DTIC
ELECTE
APR 2 1981
S D
D

81 4 2 150

AD A097022

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

This report summarizes research supported by contract F49620-80-C-0001 from the Air Force Office of Scientific Research to the University of Maryland for the period January 1, 1980 to December 31, 1980. It includes research supported in part for principal investigators Drs. Victor R. Basili, John D. Gannon, Richard Hamlet, Raymond Yeh and Marvin V. Zelkowitz, and senior personnel Drs. Mark Weiser and Pamela Zave. The references listed at the end of this report have been sent to AFOSR under separate cover.

I. Data Abstraction, Specification and Testing

I.1 PLACES Project

Work on implementing data abstractions to the PL/I compiler running under the PLACES Project continued. The language was fully implemented and included full "black box" abstract data typing where users of a type were prohibited from accessing the internal structure of a type. The implementation included several forms of assertions to be checked during program execution for program validation [Zelkowitz 80].

The implementation of these abstract types employed a novelcompiling technique where only a core set of features was added to the compiler. A built-in macro processor, as part of PLACES's first pass parser, was added to encode much of the abstraction code as macros of the basic language. This enabled the implementation to be completed relatively quickly and enabled great changes to be made "overnight" [Zelkowitz 81].

The PLACES Project was one of the first implementations of abstract data types to undergo general use. Data generated during the fall of 1980 has been collected and is now undergoing analysis. However, some features of the implementation have already become apparent and should be relevant to other such efforts (e.g., DOD Ada effort).

The overhead of using functions for all operations is quite high. However, about 60% of all such functions are simply access to selectors within the abstract type. It should be possible to automatically recognize several of these functionals and have the compiler automatically generate them. This would greatly lessen the amount of code needed to write, and should speed up execution since the compiler would be able to generate inline subroutines for these functions.

The PLACES implementation of assertions gives a good mechanism for comparing the two major specification techniques. Within an abstraction, an

assertion behaves as if it were a Hoare-type of axiom, and validation of predicate calculus proofs can be achieved. However, outside of the abstraction, the assertion is essentially an algebraic specification. Hence, there is no real distinction between both techniques. This similarity is being investigated.

I.2 DAISTS

Work begun under the previous AFOSR contract has continued on the compiler-based specification and testing system for data types, DAISTS. In an empirical study, subjects without prior experience with DAISTS were encouraged by the system to develop effective sets of test cases for their implementations. The DAISTS subjects produced implementations with fewer residual errors than subjects who used test routines provided by the experimenters without using significantly more runs. The DAISTS subjects also produced implementations that were just as good as subjects using their own test drivers, but the DAISTS subjects required fewer runs. The experiment also pointed out a hole in the coverage measures employed by DAISTS. In order to obtain better coverage, we need to add special-values testing (i.e., the constants of the implementation must appear as test points) capabilities to the system. We have also started to define the formal properties of DAISTS: the meaning of our equality function and the meaning of "successful" tests [Gannon 80].

I.3 Algebraic Specification

Although one of the advantages of the algebraic technique is that specifications are precise and formal, the fundamental definitions in this area are a source of difficulty. The crucial notion of "equality" of the specified objects is differently defined by different research workers, and this makes it difficult or impossible to compare their methods and ideas.

Mark Ardis (now at the University of Illinois) investigated the meaning of algebraic specifications, and their relationship to implementations, in [Ardis 80]. The interesting relation is the one of "correctness" which intuitively means that code is consistent with the specification. Ardis's lattices of congruences are a powerful theoretical tool for analyzing this relationship. Furthermore, he devised a specification method (Congruence Table Specifications) that is a special case of the axiom method, which is at once easier to understand and construct, and more subject to mechanical analysis. Ardis is continuing and extending this work started at Maryland in his new position at Illinois.

I.4 Testing Theory

"Reliability" is a property of program tests that connects testing with program correctness: when a program passes a reliable test, it is necessarily correct. Two aspects of this idea require further investigation. One is the connection with coverage. Intuitively, a test is made more reliable if it "covers" a program in some structurally-defined sense. However, the technical definitions do not capture this intuitive idea--both path and data coverage measures do not lead to reliability. The second aspect of reliability being investigated is that of "determining" programs rather than proving them correct. Tests can be devised (based on coverage, in fact) that guarantee a program to be unique in that no other can pass the test (including coverage criteria). However, except for the tested behavior, the (necessarily unique) specification of the program may be unknown. These ideas are investigated in [Hamlet 80].

Larry Morell is investigating fundamental new approaches to testing theory. Two promising ideas are:

1) Probalistic testing. The difficulty with tests is that they are small samples drawn from large spaces; thus passing a given test may not significantly increase the probability of future success, because the portion of the potential test space examined is negligible. A useful probabilistic theory must attain a multiplication effect of many orders of magnitude. Although such a theory may be very hard to discover, recent successes in number theory (probabilistic determination of primes, for example) are suggestive.

2) Combined testing and proving. Past attempts to combine the effectiveness of tests with the power of proofs have concentrated on testing a few cases of the general invariant assertions for a program. Morell's idea is that instead we seek to write assertions that are not conventional invariants, but rather "enabling" assertions for the usefulness of tests. These assertions would probably take the form of asserting the existence of program limits and restrictions, and if the assertion were proved, it would guarantee that certain tests would then prove the program correct. The advantage would be in shifting some of the burden from proof to test, since the "enabling" assertions would be simple.

II. Program Correctness

Work continued in analyzing the theoretical aspects of functional correctness and comparing it with other approaches [Dunlop 80]. The highlights of

this report are as follows:

1) A function derivation/verification proof strategy based on prime program decomposition is presented.
2) An implication of the functional theory for the derivation of loop invariants is discussed.
3) Functional correctness for loops is described as a specialization of the inductive assertion correctness technique.
4) Functional correctness is compared and contrasted with subgoal induction.
5) The problem of synthesizing a loop invariant and an intended loop function for the initialized loop program form are shown to be equivalent.

In addition, work was begun on a new correctness strategy for iterative programs. The technique is based on a generalization of the WHILE loop verification rule of functional correctness and does not rely on prime program decomposition. A chief advantage to this verification strategy is that it identifies circumstances in which efficient proofs are possible. These proofs are accomplished by employing the program specification to ascertain the necessary properties of contained loops. In other cases, the methodology proposes a heuristic for synthesizing intended loop functions which seems to work well on uniformly behaving loops. A report describing this work is currently in preparation.

III. Complexity Metrics

A family of structural complexity metrics has been developed which contains many control and size metrics from the current literature [Basili 80]. These include statement count and cyclomatic complexity. The family allows for the use of factors such as size, nesting level, and type of control structures in the computation of complexity. This family has been used in further investigations of the set of student compiler projects described in [Basili 81B]. The effort has been mainly concerned with the relationships between some metrics in the family and the number of program changes made during the development of the projects.

It was found that the relationship is stronger for projects developed by a single individual than for projects developed by teams. Furthermore, the relationship was found to be more a straight line than an exponential curve

and the slopes of the lines for the different individuals varied widely. It is believed that this provides the beginning of a measure of how well a given programmer responds when faced with a given level of complexity. More results and statistical analysis may be found in [Basili 81A].

The compiler projects have been subject to further experimentation by having some of them modified by another group of students. No results of this additional experimentation is yet available.

IV. Requirements and Specification

The objective of this research was to develop an integrated, practical methodology. We had already done work on more specialized areas. In 1980, while we were continuing the work, we were also in the process of integrating these different methodologies. Our work can be broken down into the following major sub-areas:

IV.1 Conceptual Modeling

This is a continuation of the work done previously using semantic network as the descriptive notation for constructing the conceptual model. In order to enhance the methodology, we propose to study the work breakdown structure of the requirements analysis phase. A 3-leveled modeling formalism has been proposed: abstract task modeling for organization investigation, conceptual system modeling, and detailed requirements flow representation. Semantic net notation was extended to cover the control aspects as well.

IV.2 Operation Requirements Specification

Several new requirements examples increased our confidence in language's ability to deal with the range of embedded systems requirements, especially performance (timing and reliability) requirements. Therefore, the language syntax was fully defined in terms of an LALR grammar. We have investigated the relationship between PAISLEY verifications and data-oriented requirements models, and discovered that they are basically compatible because the collective states of processes in a PAISLEY specification are such a data-oriented model, with restrictions entirely justifiable from the viewpoint of embedded systems. Zave also began work on the problem of creating hierarchically-structured requirements specifications in PAISLEY, with very abstract versions useful at the earliest "conceptual" stages. [Yeh 80B]

IV.3 Specification of the Concurrent Processes and Communication Protocols

A high-level specification language for concurrent and distributed system is being developed. The language is "event-based" as contrast to other state-machine-based specification techniques. [Zave 80]

V. Programming Environments

Iterative enhancement, a technique developed by Basili, is a method for developing software by gradually enhancing a working program until it has met certain specifications. This method offers opportunities for new kinds of software development tools through high-quality user feedback, bootstrapping, assuring consistency across iterations, and a "programmer's apprentice." One tool in particular, program slicing, as developed by Weiser, is considered in detail. Starting from a subset of a program's behavior, slicing reduces that program to a minimal form which still produces that behavior. The reduced program, called a "slice," is an independent program sufficient to faithfully represent the original program within the domain of the specified subset of behavior. Slicing-based development tools are proposed for eliminating superfluous code and for identifying code with certain behaviors [Weiser 81].

References

[Ardis 80] M. A. Ardis, Data abstraction transformations, Ph.D. dissertation TR-925, Department of Computer Science, University of Maryland, 1980

[Basili 80] V. R. Basili and D. H. Hutchens, "A Study of A Family of Structural Complexity Metrics," Proc. ACM-NBS Nineteenth Annual Technical Symposium: Pathways to System Integrity, Gaithersburg, MD., June 1980, pp. 13-15

[Basili 81A] V. R. Basili and D. H. Hutchens, "Analyzing A Syntactic Family of Complexity Metrics," ACM/SIGSOFT Software Engineering Symposium on Tool and Methodology Evaluation, Pingree Park, Colorado, June 9-11, 1981

[Basili 81B] V. R. Basili and R. W. Reiter, "A Controlled Experiment Quantitatively Comparing Software Development Approaches," IEEE Transactions on Software Engineering, 1981, to appear

[Dunlop 80] D. D. Dunlop and V. R. Basili, "A Comparative Analysis of Functional Correctness," Technical Report TR-921, University of Maryland, Computer Science, 1980

[Gannon 80B] J. D. Gannon, P. R. McMullin, R. G. Hamlet, Data Abstraction, Implementation, Specification, and Testing (submitted for publication) 1980

[Hamlet 80] R. G. Hamlet, Reliability Theory of Testing (submitted for publication) 1980

[Miyamoto 81] I. Miyamoto and R. T. Yeh, "A Software Requirements Analysis and Definition Methodology for Business Data Processing," to appear NCC Proceedings

[Weiser 81] M. Weiser, Towards an iterative enhancement software development environment, 14th Hawaii International Conference on Systems Science, 1981

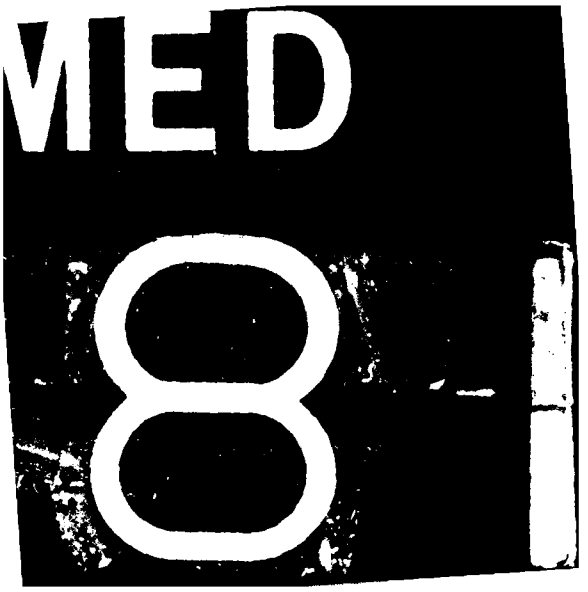
[Yeh 80A] R. T. Yeh, P. Zave, A. P. Conn, G. E. Cole, Jr., "Software Requirements: A Report on the State of the Art," TR-949, October 1980, Computer Science Technical Report Series, University of Maryland

[Yeh 80B] R. T. Yeh, P. Zave, "Specifying Software Requirements," Proceedings IEEE, September 1980

[Zave 80] P. Zave and R. T. Yeh, "Executable Software Requirements for Embedded Systems," to appear Proceedings 5th ICSE

[Zelkowitz 80] M. V. Zelkowitz and J. R. Lyle, Implementation of program specifications, IEEE COMPSAC, Chicago, Ill., 1980

[Zelkowitz 81] M. V. Zelkowitz and J. R. Lyle, Development of program enhancements (submitted) 1981


MED
8